Trabalho 5

Introdução ao Processamento Digital de Imagem (MC920/MO443)
Professor: Hélio Pedrini

Vinicius Couto Espindola | RA: 188115

7 de Julho de 2020

1 Introdução

Grande gerenciadores de fotos apresentam diversas ferramentas as quais possibilitam uma série de classificações e transformações sobre images: agrupamento de imagens, reconhecimento facial, colagem panorâmica, dentre outras. Muitas destas são baseadas em ferramentas as quais visam encontrar pontos semelhantes entre imagens que permitam traçar alguma relação entre elas, seja esta dimensional, qualitativa ou quantitativa. O foco deste trabalho será voltado à operação denominada de "image stitching", a qual extraí pontos em comum de duas imagens para unir-lás em uma, tal operação é útil quando precisamos "costurar" múltiplas imagens juntas, como fotos panorâmicas por exemplo.

2 Execução

O programa requer as seguintes bibliotecas: *OpenCV*, *OpenCV Contrib* e *NumPy*. Dado que alguns do algoritmos utilizados neste trabalho são patenteados (SIFT e SURF), utilizaremos uma versão mais antiga do *OpenCV* e o pacote de módulos extras *OpenCV contrib*. Ambos pacotes devem estar na versão **3.4.2.16**. O programa pode tanto exibir a imagem de saída, quanto salvar o resultado em um arquivo. Todo os resultados produzidos e utilizados encontram-se na pasta "*./outputs*". As imagens de entrada utilizadas estão na pasta "*./inputs*". Vale ressaltar que a ordem nas quais as imagens são atribuídas para o programa é relevante, caso um transformação pareça errada, tente inverter a ordem na qual as imagens são definidas na linha de comando. As imagens de entradas em "*./inputs*" estão identificadas como **a** e **b**, as quais correspondem aos argumentos *imageA* e *imageB* da linha de comando.

$$assignment_5.py\ [-h]\ [-t\ [threshold]]\ [-o\ output]\ [-v]\ imageA\ imageB\ Descriptor$$

imageA: *Path* da imagem que servirá como plano de destino da transformação.
imageB: *Path* da imagem que terá seu plano transformado ao plano da *imageA*.
Descriptor: Deve ser uma string dentre *"sift", "surf", "brisk" e "orb"* correspondendo ao extrator e descritor de *features* que será utilizado.
-t threshold: Define o erro máximo entre dois descritores (*float*) para que estes possam ser considerados equivalentes entre duas imagens distintas. Este parâmetro é opcional e, caso não seja definido pelo usuário, o valor padrão de 0.1 será utilizado.
Nota: Caso a margem de erro seja muito pequena, é possível que o programa não execute, uma vez que a transformação das imagens requer pelo menos quatro pontos para ser realizada.
-v ou –verbose: Caso seja utilizada, o pareamento de features entre as imagens assim como imagem final serão exibidos na tela.
-o output-file: *Path* onde a imagem de saída deve ser salva.
Nota: Caso não seja definido um *output-file*, a imagem será exibida na tela. Se for definido, a imagem de saída será salva apenas (a não ser que a flag **-v** seja usada).

3 Soluções

Opencv e o pacote extra *OpenCV Contrib* apresentam diversas ferramentas para realizar as operações necessárias:

1. Identificação e descrição de features serão feitas através das classes: *cv2.xfeatures2d.SIFT_ create(), cv2.xfeatures2d.SURF_ create(), cv2.BRISK_ create() e cv2.ORB_ create()*
2. Para parear os descritores em duplas usaremos o *cv2.BFMatcher()* que realiza o agrupamento dos k pontos mais similares por *Brute Force*
3. A matriz de homografia será calculada pela função *cv2.findHomography()* a qual permite o uso do método *RANSAC* para estimar a matriz.
4. A transformação do plano da imagem será feita pela função *cv2.warpPerspective()*.

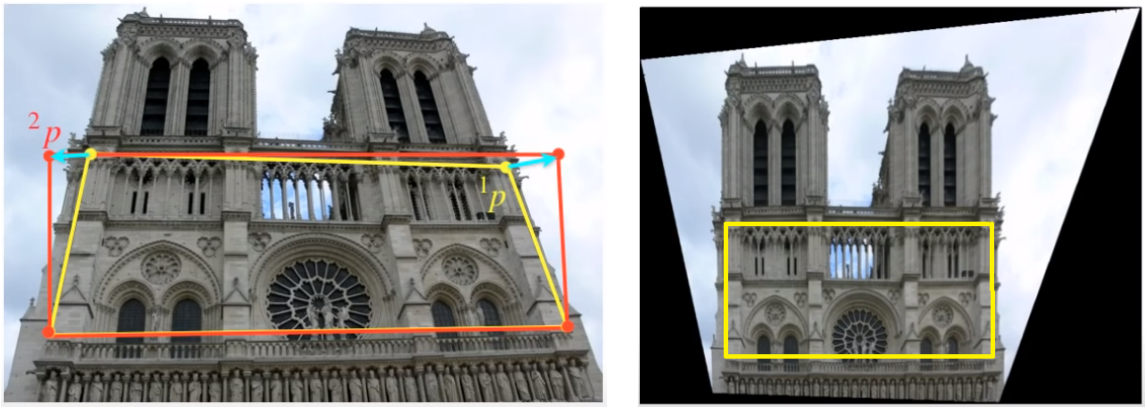
3.1 Pareamento de Pontos de Interesse

A maioria das funções utilizadas realizam tanto a identificação quanto a representação dos pontos de interesses da imagens em questão. Cada uma das funções enunciadas realizam a tarefa de encontrar as *features* das imagens e descrevê-las de forma que estes mesmos pontos possam ser detectados em imagens de diferentes perspectivas e escalas. O *SIFT* por exemplo consegue extrair estes pontos de forma que as representações sejam independentes da orientação e escala da imagem. Assim que realizamos a chamada destas funções, temos duas variáveis de retorno as quais indicam a localização das features (*keypoints*) e os descritores destes respectivos pontos. Com estes valores, podemos comparar os pontos pelos descritores e então associar os pixels cujos descritores são mais similares entre as imagens através dos *keypoints*. Por fim, extraímos os pixels cujos descritores apresentam erro menor que o limite definido pelo usuário, e então seguimos para o cálculo da matriz de homografia.

Figure 1: Exmplo de extração e descrição de features com o *SIFT* (Source)

3.1.1 Matriz de Homografia e *Warping*

Com quatro pontos de cada um dos dois planos definidos pelas imagens podemos passar a aplicar a transformação *imagemB* para alinhá-la com a *imagemA*. Esta transformação é realizada pelo conceito de homografia. De forma grosseiramente simplificada, podemos intuir sobre este processo como o deslocamento de um conjunto de pontos de uma perspectiva para outra, de forma que as mudanças de proporção sejam respeitadas ao máximo. O propósito da matriz de homografia é descrever o deslocamento que os pontos de uma imagem precisam realizar para que os *keypoints* pareados entre as imagens coincidam posicionamento entre as imagens após à transformação. O processo de *warping* realizado através da matriz de homografia está exemplificado na **Figura 2**.



(a) Trasnformação a Ser Feita (b) Resultado da Transformação

Figure 2: Exemplo de trasformação por homografia

A seleção adequada da margem de erro entre os pontos é essencial para esta etapa, uma vez que, caso os pontos associados não representem seus equivalentes entre as duas imagens, a transformação realizada poder ficar distorcidas. Mesmo que os pontos de interesse fiquem relativamente próximos, temos que erros pequenos podem alterar a inclinação e rotação da imagem no plano de destino. Um exemplo de um caso onde poucos pontos muitos próximos uns dos outros são encontrados, o que resulta em uma transformação imprecisa da imagem pode ser encontrado na **Figura 3**. Observa-se que, apesar dos pontos de interesse entre as imagens estarem relativamente próximos, temos que a inclinação da *imagemB* após a transformação está significativamente errada.

(a) Poucos Pareamentos Encontrados (b) Transformação Distorcida

Figure 3: Exemplo de trasformação distorcida

3.1.2 Pósprocessamento

Podemos realizar uma série de operações após a união entre as imagens. É relativamente comum, por exemplo, que as imagens apresente diferenças de contraste e resolução, o que pode ser ajustado através de filtros após a união. No caso, o único pósprocessamento aplicado foi a remoção do excesso das bordas pretas que remanescem após a união das imagens.

3.2 Limitações

A principal restrição deste processo é o fato que nós precisamos definir as associações entre *imagemA* e *imagemB* corretamente, caso contrário, a transformação pode sair completamente incorreta. Em relação a restrições de performance e numéricas o código é bem robusto, principalmente por utilizar métodos já trabalhados extensivamente no *OpenCV*. Temos também algumas restrições legais para os algoritmos *SIFT* e *SURF* que são patenteados, no entanto este não são restritos para aplicações acadêmicas.

4 Resultados

Foram testados quatro métodos de reconhecimento e descrição de *features* de imagens. O programa também foi testado com diferentes situações de posicionamento bidimensional das imagens a serem costuradas. Os resultados obtidos com o costuramento das imagens, assim como o pareamento dos pontos das imagens de acordo com os descritores mais semelhantes entre elas, estão ilustrados nas figuras abaixo.

Figure 4: Junção de imagem transladada e rotacionada usando *SIFT* ($threshold = 0.1$)

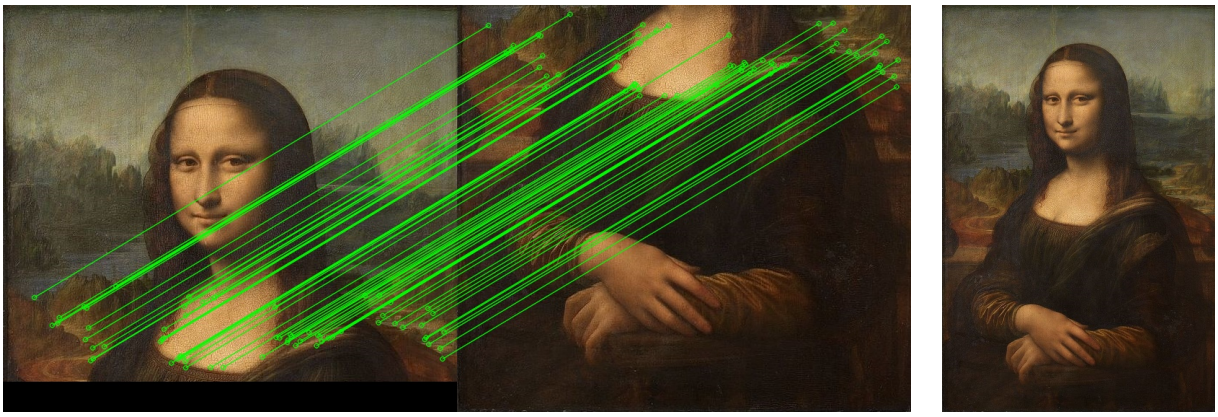


Figure 5: Junção vertical de imagens usando *SURF* ($threshold = 0.1$)

Os métodos *SIFT* e *SURF* apresentaram bons resultados, conseguindo detectar múltiplos pares de *keypoints* com *thresholds* baixos, o que acarreta em junções de maior qualidade. Observando as **Figuras 4 e 5** temos que os resultados não apresentam nenhum tipo de distorção. No caso da **Figura 4**, temos que as margens da junção apresentam um contorno branco bem pouco visível, enquanto a *Mona Lisa* tem sua junção quase que imperceptível. Ambos métodos também foram rápidos para encontrar e realizar a descrição das features.

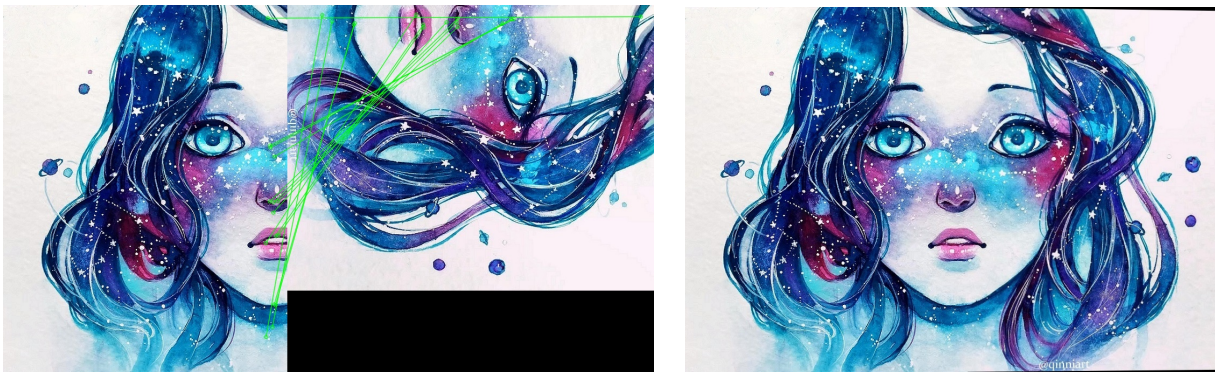


Figure 6: Junção horizontal de imagem rotacionada usando *BRISK* ($threshold = 0.2$)
Copyrights: Starred Freckles, Qinni

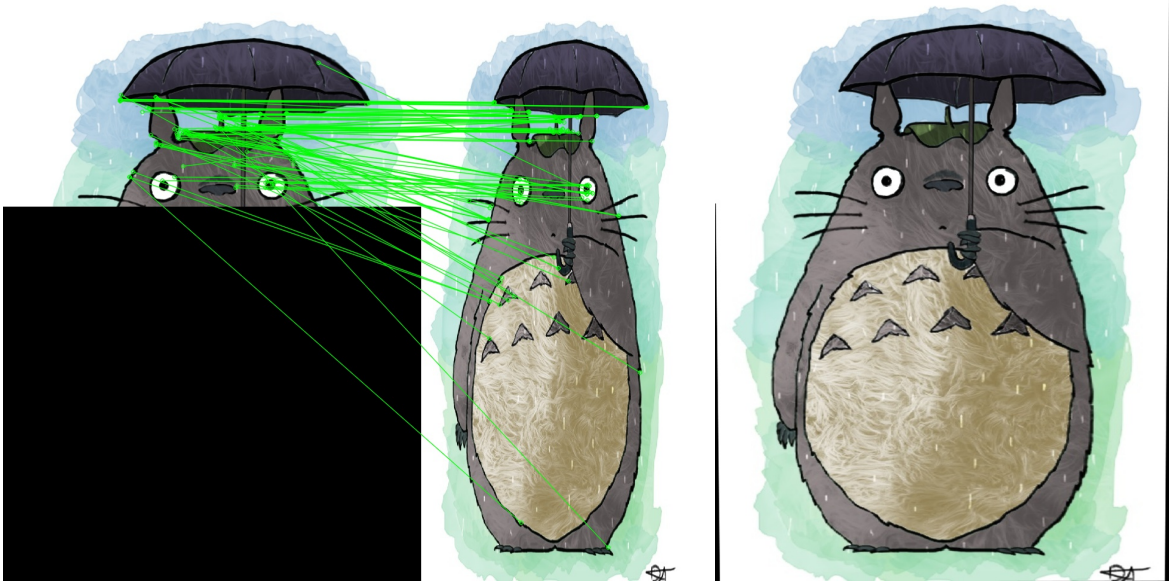


Figure 7: Junção horizontal de imagem distorcida usando *ORB* ($threshold = 0.9$)

Observa-se que os métodos *BRISK* e o *ORB*, apesar de apresentarem resultados relativamente bons, não alcançaram a mesma qualidade que os dois métodos anteriores. Nota-se que ambas as **Figuras 6 e 7** não realizam a transformação tão precisamente quanto os métodos anteriores. Isso deve-se principalmente ao fato que ambos utilizaram erros relativamente altos para o pareamento dos *keypoints*, o que possibilita pareamentos de pontos imprecisos que, consequentemente, geram distorção no processo de *warping*. Vale ressaltar também que o *BRISK* foi o algoritmo mais lentos de todos, os outros apresentaram tempo de execução similares e curtos.

5 Discussão e Conclusão

O costuramento de uma imagem é um processo complexo que envolve múltiplas etapas complexas. Temos que a extração de *features* da imagem é um passo essencial para múltiplos algoritmos, pois possibilita a associação de imagens que representam o mesmo objeto mas em perspectivas e escalas diferentes. Esta etapa por sí só já apresenta uma enorme quantidade de conteúdo a ser considerado, como definir quais pixels de fato representam features e ainda por cima definir como representá-los de forma independente escala e orientação. O segundo passo notável deste processo é a transformação do plano da imagem por homografia. Esta operação possibilita a transformação da informação contida na imagem de forma que ela represente uma perspectiva nova. Certamente essa transformação não é perfeita, uma vez que em muitos casos esta se baseia na extrapolação da informação disponível, no entanto, para casos bidimensionais, costuma ser mais do que suficiente para gerar costuras quase perfeitas, uma vez que tem toda informação necessária disponível. Transformações que mudam perspectiva no entanto requerem a informação tridimensional do objeto da foto para serem perfeitas. Os resultados mostram que as ferramentas utilizadas são extremamente poderosas e promissoras, o que é de se esperar dado que tais operações são utilizadas em inúmeros softwares visuais atualmente.

References

[1] R.C. Gonzalez. *Digital Image Processing.* Prentice Hall, 2007.